REMARKS

In the Final Office Action mailed February 7, 2008, the Examiner maintained the rejection of claims 1, 14, 19, and 25 under 35 U.S.C. § 103 as being unpatentable over U.S. Patent No. 5,995,103 to Ashe ("Ashe") in combination with U.S. Patent Number 5,920,313 to Diedrichsen *et al.* ("Diedrichsen"). Neither Ashe nor Diedrichsen, separately or in combination, teach or suggest grouping GUI windows in which independent applications are running, and simultaneously altering the z-order of all windows in the group when one is selected.

In Response to Arguments, the Examiner cited to col. 1, lines 63-65 of Diedrichsen, which states, "Different applications can also be organized into groups of applications, each of which are related by function." This statement appears in the Background section of Diedrichsen's written description, before the concept of z-ordering is introduced, and certainly before any idea of linking parent/child windows and altering the z-order of such windows together is introduced. The statement is not remotely related to Diedrichsen's inventions of linking parent/child windows for visual identification or of altering the z-order of parent/child windows together.

Diedrichsen discloses precisely one mechanism for linking windows to alter their z-order together: when an application spawns a child, it stores a pointer to the child in a private list. This is described at col. 8, liens 23-34:

> FIG. 6 shows the data structure used to implement the set membership relation between user interface objects. Every time an application creates a child window 620 from an object 610, it stores a pointer to that object in a private list 630 of child windows; equally, when the child window 620 is created, it is passed a pointer 640 to its parent window. If the child window 620 is destroyed, it will use the pointer 640 to its parent window 610 to call a method removing itself from the list of child window pointers 630. Hence the parent window 610 maintains an up-to-date list of pointers 630 to all its child windows.

The parent uses this list to cycle through all its child windows when it is selected, as depicted in the loop 740-745-750 of Figure 7b. If a child window is selected, it detects whether it has a

parent, and if so, cycles through all of that parent's other child windows, as depicted in the loop 755-760-765-770 of Figure 7b. Diedrichsen discloses no other mechanism for altering the z-order of a group of windows together. It is impossible for Diedrichsen to simultaneously alter the z-order of a group of windows running independent applications. The independent applications would have no knowledge of each other, since the private list 630 is only created and populated when an application spawns a child window. This is true even for "[d]ifferent applications . . . which are related by function."

Furthermore, even if the cited statement appeared in the heart of Diedrichsen's explanation of his invention – indicating some relevance thereto – it would not teach or suggest Applicants' claimed invention. First, the statement says only that different applications can be organized into groups. It is completely silent as to even a hint that such grouped applications may be linked in z-order manipulations. Second, even if the statement were somehow construed to offer such teaching, it is expressly limited to applications "which are related by function." Applications which are related by function are not independent applications, by definition. Applicants' specification provides the example of a web browser, email client, and word processor being grouped together. These applications are completely **unrelated** by function. Diedrichsen's teaching is limited to altering the z-order of windows running applications that are related as parent and child.

The Examiner then asserted that "Ashe discloses examples of different independent applications such as word processing and a spreadsheet application grouped together by priority number," citing to col. 2, lines 10-15. That passage – again, appearing in the Background section – is part of an explanation of the concept of z-ordering. It states,

> A window layer's priority class defines where in the z-order the window layer can be displayed. For example, a window layer of priority class "2", e.g. a screen saver, will always appear in front of a window layer of priority class "3", e.g. an application program, while multiple window layers of priority class "3", e.g. a word processing application and a spreadsheet application, can overlie each other in the z-order.

This passage does not disclose any grouping at all. Rather, it states only that multiple applications may have the same priority – an entirely unremarkable concept in the computer arts. Typically, **all** user applications running in a GUI environment will be in the same priority class, and will "overlie each other in the z-order." Calling all applications running in a GUI environment a "group" is a trivial application of the concept of grouping, and one that is nonsensical in the context of simultaneous z-order manipulation.

Ashe explicitly states that multiple applications having the same priority class "can overlie each other in the z-order." If anything, this teaches away from the concept of grouping independent applications, and changing their z-order together, since the motivation for a user doing so is almost always to prevent the windows of interest overlying each other in the display. Ashe's teaching is limited to selecting a subset of windows spawned by a single application, such as toolbars, and raising the z-order of the subset of windows, rather than all such windows associated with the application, when the main program window is selected.

One of ordinary skill in the art, having Ashe and Diedrichsen before him, would be enabled to group together, for z-order manipulation, an application window and windows spawned by that application. Ashe teaches group z-order control of a subset of the toolbar windows that are spawned as part of the same application. Diedrichsen teaches group z-order control of windows related as parent and child. In both cases, a single application spawns all of the windows that join it in group z-order manipulation. Accordingly, the originating window has explicit knowledge of all other windows in the group (or that may be selectively included in a group). In the case of independent applications, no application has any knowledge of even the existence of a different independent application, much less the ability to alter or influence the other window's z-order.

Neither Ashe nor Diedrichsen, alone or in combination, teach or suggest the concept of grouping windows running independent applications for the purpose of simultaneous z-order

manipulation. Nor would one of ordinary skill in the art, studying Ashe and Diedrichsen, have any idea how to proceed to implement such an idea. A key feature of GUI operating systems is that each independent application, running in a separate window, has no knowledge of, or access to, any other application running in any other window. Ashe and Diedrichsen teach that, in special cases where one application spawns other windows (either as toolbars or as child applications), that knowledge of the other windows may be utilized to control the z-order of at least some of the windows together. Nothing about these disclosures remotely hints at the group z-order control of independent applications.

For at least the reason that the combination of Ashe nor Diedrichsen fails to teach or suggest the group z-order control of windows running independent applications, the § 103 rejections of claims 1, 14, 19, and 25 are improper and must be withdrawn. All dependent claims include all limitations of their respective parent claim(s), and thus also define patentable nonobviousness over the art of record.

All pending claims are in condition for allowance, which prompt action is hereby respectfully requested.

Respectfully submitted,

COATS & BENNETT, P.L.L.C.

Dated: March 3, 2008

Edward H. Green, III
Registration No.: 42,604

1400 Crescent Green, Suite 300
Cary, NC 27518

Telephone: (919) 854-1844
Facsimile: (919) 854-2084